加速法とその漸化式表現(非線形可積分系による応用解析)

降旗, 大介

数理解析研究所講究録
第 889 巻 1994 年 26-36

加速法とその漸化式表現

東京大学工学部物理工学科 降旗 大介 (Daisuke Furihata)

1 はじめに

数値計算において収束する数列を扱うケースは非常に多い。一般に反復法とよばれる数値解法によって方程式の解を求める時などもこれに該当する。

数学の立場からは収束するかどうかが基本的な問題であるが、数値計算の立場からは数列の収束の速度がもっとも重要な問題となる。というのも、収束する速度により数値計算の効率が大きく異なるだけではなく、ある程度以上に収束速度が遅いと実際問題として数値計算ができないからである。また、数列そのものの計算が非常に大変な場合は、なるべく少ない要素数で収束先の良い推定値を得たいという要求にたいする必要条件が収束速度の改善なのである。

こうした問題に対し、加速法は収束数列の収束速度を改善するために開発されてきた数値計算の手段である。(その概要や現状についての報告は [2, 4] などに詳しい。)

加速法の研究は一般には数列の変換における性質の変化を研究するという形をとるが、このやり方はなぜそのような変換を用いるとよいのかが分かりにくい。そこで、本研究では特に加速法を「数列の補間・補外における優れたアルゴリズム」という視点でとらえてみることにより、加速法の一部を分かりやすい形で紹介することを試みるものである。

2 加速法と補間・補外の関係

加速法は補間・補外アルゴリズムの一種であり、次のように示されると考える。

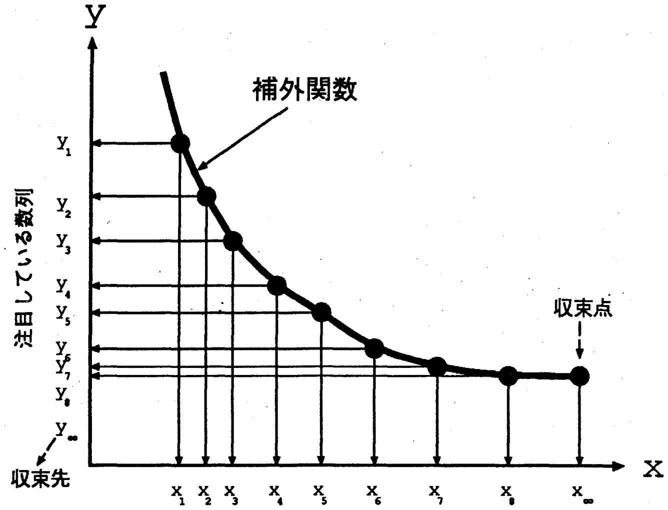

図 1: 補間・補外と加速法の関係

対象となっている数列 $\{y_i\}_{i\in\Lambda}$ を標本点 $\{x_i\}_{i\in\Lambda}$ におけるある関数 $y(x)$ の関数値だとみなすことにより、$x \longrightarrow x_\infty$ への補外値 $y(x_\infty)$ を求めると考えるのである。基本的に、加速法は収束数列の収束の様子をある程度仮定して行なわれるため、収束数列がある補外関数の標本値とみなせるというのは無理な前提ではない。このときの補外を行うアルゴリズムのうち、特に計算速度を改良したものを一般に加速法と呼ぶとみなすのである。なお、全ての加速法をこの考え方でとらえられるという保証はないが、多くはこの考え方によって理解できる。

このとき、$n+1$ 個のデータを用いる補外式

$$y_{i+n}^{(n)} \equiv f_n(y_i, y_{i+1}, \cdots, y_{i+n}) \tag{2.1}$$

を考えると、より真の値に近い補外を行なうには

(A). 利用するデータ数を増やす $\Longleftrightarrow$ n を大きくする

(B). 利用するデータをより真の値に近いものにする $\Longleftrightarrow$ i を大きくする

の 2 つの方向があることが分かる。

なるべく小さい i を用いて良い補外値を得るのが加速法の本来の立場であるから、i を大きくするというのはあまり得策ではない。

しかし、n が大きい場合には上の補外式 f_n の計算量は補外アルゴリズムによっては非常に大きくなる。補外式が行列式を用いて表現される場合などがそれに当たり、その場合はデータ量 n に対して計算量は $n!$ で増大する。

これに対し、$y_{i+n}^{(n)}$ 同士の間に成立する漸化式を発見することにより、直接 f_n を用いて計算することを避けるという方法がある。これから紹介する加速法はこの方法に基づいていると解釈されるものである。

補外が補間の一つの具体形であることに対応して、これらの加速法も

(A). 補間式に漸化式が存在し、その漸化式に $x \leftarrow x_\infty$ と値を代入するとその加速法となる

(B). 具体的に補外式を表すことで漸化式が見つかり、その加速法となる

という 2 つのタイプに分けられる。なお、補外式に漸化式が存在する経路の方は、その補間式段階での漸化式を知らないためにこういう分類をしている可能性がある。

3 補間式間に漸化式が存在する場合

補間式そのものに漸化式が存在し、その漸化式に具体的に値を入れることで加速法が得られるという例として、多項式補間および有理式補間を用いて補間を行なってから加速法を得ている場合を紹介する。

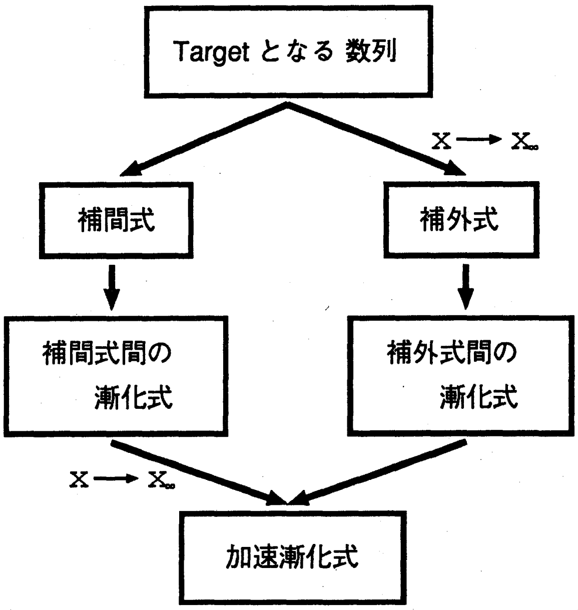

図 2: 加速法の漸化式表現を導く 2 つの経路

3.1 多項式補間を用いる場合 : Neville algorithm

表 1 に多項式補間に基づく加速法の概要を表にしてある。

加速したい収束数列 $\{y_j\}_{j=1}^{\infty}$ が適当に標本点 $\{x_j\}_{j=1}^{\infty} : \lim_{j\to\infty} x_j = 0$ をとると、十分おおきな j に対して標本点座標の多項式

$$y(x) = a_0 + a_1 x + a_2 x^2 + \cdots \tag{3.1}$$

で補間できると想定できる場合、n 点の標本点 $\{x_j\}_{j=1}^{n}$ から定まる $n-1$ 次式多項式補間関数を $P[x_1, \cdots, x_n](x)$ とすると、

$$P[x_1, \cdots, x_n; x_j, x_k](x) = \frac{(x - x_j)P[x_1, \cdots, x_n; x_k](x) - (x - x_k)P[x_1, \cdots, x_n; x_j](x)}{x_k - x_j} \tag{3.2}$$

が成立するため(この関係式そのものが既に漸化式の形をしていますが)、補間式間に Neville algorithm [7, pp.29–]

$$P[x_1, \cdots, x_{n+1}](x) = \frac{(x - x_{n+1})P[x_1, \cdots, x_n](x) - (x - x_n)P[x_2, \cdots, x_{n+1}](x)}{x_n - x_{n+1}} \tag{3.3}$$

が考えられる。

欲しいのは $j \to \infty$ の補外値であるから、この場合は $x \to 0$ の場合に対応し、n 点の標本点 $\{x_j\}_{j=1}^{n}$ から定まる a_0 の補外値 (y_∞ に対応する) を $C[x_1, \cdots, x_n]$ とすると、加速法である **Neville algorithm**

$$C[x_1, \cdots, x_{n+1}] = \frac{-x_{n+1}C[x_1, \cdots, x_n] + x_n C[x_2, \cdots, x_{n+1}]}{x_n - x_{n+1}} \tag{3.4}$$

補間・補外関数	多項式 $y(x) = a_0 + a_1x + a_2x^2 + \cdots$ (a_j: 未知)
補間式	$n-1$ 次式 (自由度 n) $$P[x_1,\cdots,x_n](x) = \sum_{j=1}^{n}\left(y_j \frac{\prod_{k\neq j}(x-x_k)}{\prod_{k\neq j}(x_j-x_k)}\right)$$
漸化式	Neville algorithm $$P[x_1,\cdots,x_{n+1}](x) = \frac{(x-x_{n+1})P[x_1,\cdots,x_n](x)-(x-x_n)P[x_2,\cdots,x_{n+1}](x)}{x_n-x_{n+1}}$$
加速法	a_0 への加速 : **Neville algorithm** ($x \to 0$) $$C[x_1,\cdots,x_{n+1}] = \frac{-x_{n+1}C[x_1,\cdots,x_n]+x_nC[x_2,\cdots,x_{n+1}]}{x_n-x_{n+1}}$$

表 1: 多項式補間に基づく加速法

が得られる。(ただし、$C[x_j] = y_j$ である。)

なお、この Neville algorithm の他に同様の Aitken algorithm もあるが、これも (3.2) に基づくものであり、基本的には同じものと考えられるため省略する。

3.2 有理式補間を用いる場合 : BS–加速、ρ–algorithm

補間・補外関数	有理式 $y(x) = \dfrac{a_0+a_1x+\cdots+a_mx^m}{b_0+b_1x+\cdots+b_nx^n}$ (a_j, b_j : 未知)
補間式	m 次式/n 次式 (自由度 $m+n+1$) : $R[x_1,\cdots,x_{m+n+1}](x_j) = y_j$
漸化式	BS algorithm
加速法	**BS 加速** : $\frac{a_0}{b_0}$ への加速 ($x \to 0$)
	ρ–algorithm : ($n = m$ として) $\frac{a_m}{b_m}$ への加速 ($x \to \infty$)

表 2: 有理式補間に基づく加速法

表 2に有理式補間に基づく加速法の概要を表にしてある。

この場合も前節と同様に収束列が適当に標本点 $\{x_j\}_{j=1}^{\infty} : \lim_{j\to\infty} x_j = 0$ をとると、十分おおきな j に対して標本点座標の有理式

$$y(x) = \frac{a_0+a_1x+\cdots+a_mx^m}{b_0+b_1x+\cdots+b_nx^n} \tag{3.5}$$

で補間できると想定する。このとき、$m+n+1$ 点の標本点 $\{x_1,\cdots,x_{m+n+1}\}$ から定まる有理式補間関数

$$R^{m,n}[x_1,\cdots,x_{m+n+1}](x) \equiv \frac{P^{m,n}[x_1,\cdots,x_{m+n+1}](x)}{Q^{m,n}[x_1,\cdots,x_{m+n+1}](x)} = \frac{a_0+a_1x+\cdots+a_mx^m}{b_0+b_1x+\cdots+b_nx^n} \tag{3.6}$$

(ただし $P^{m,n}(x)$ は m 次多項式、$Q^{m,n}(x)$ は n 次多項式であるとする)
に対して、$S \equiv \{x_1, \cdots, x_{m+n}\}$ として次の漸化式が成立する。

$$R^{m+1,n}[S;x_l,x_k](x) = R^{m,n}[S;x_l](x) + \frac{R^{m,n}[S;x_k](x) - R^{m,n}[S;x_l](x)}{1 - \left(\frac{x-x_k}{x-x_l}\right)\frac{R^{m,n}[S;x_k](x)-R^{m,n-1}[S](x)}{R^{m,n}[S;x_l](x)-R^{m,n-1}[S](x)}} \tag{3.7}$$

$$R^{m,n+1}[S;x_l,x_k](x) = R^{m,n}[S;x_l](x) + \frac{R^{m,n}[S;x_k](x) - R^{m,n}[S;x_l](x)}{1 - \left(\frac{x-x_k}{x-x_l}\right)\frac{R^{m,n}[S;x_k](x)-R^{m-1,n}[S](x)}{R^{m,n}[S;x_l](x)-R^{m-1,n}[S](x)}} \tag{3.8}$$

そこで、

$$R_j^{m,n}(x) \equiv R^{m,n}[x_j, \cdots, x_{j+m+n}](x) \tag{3.9}$$

とおき、この補間有理式における次のような BS–algorithm が得られる。

$$R_j^{m+1,n} = R_j^{m,n} + \frac{R_{j+1}^{m,n} - R_j^{m,n}}{1 - \left(\frac{x-x_{j+m+n+1}}{x-x_j}\right)\frac{R_{j+1}^{m,n}-R_{j+1}^{m,n-1}}{R_j^{m,n}-R_{j+1}^{m,n-1}}} \tag{3.10}$$

$$R_j^{m,n+1} = R_j^{m,n} + \frac{R_{j+1}^{m,n} - R_j^{m,n}}{1 - \left(\frac{x-x_{j+m+n+1}}{x-x_j}\right)\frac{R_{j+1}^{m,n}-R_{j+1}^{m-1,n}}{R_j^{m,n}-R_{j+1}^{m-1,n}}} \tag{3.11}$$

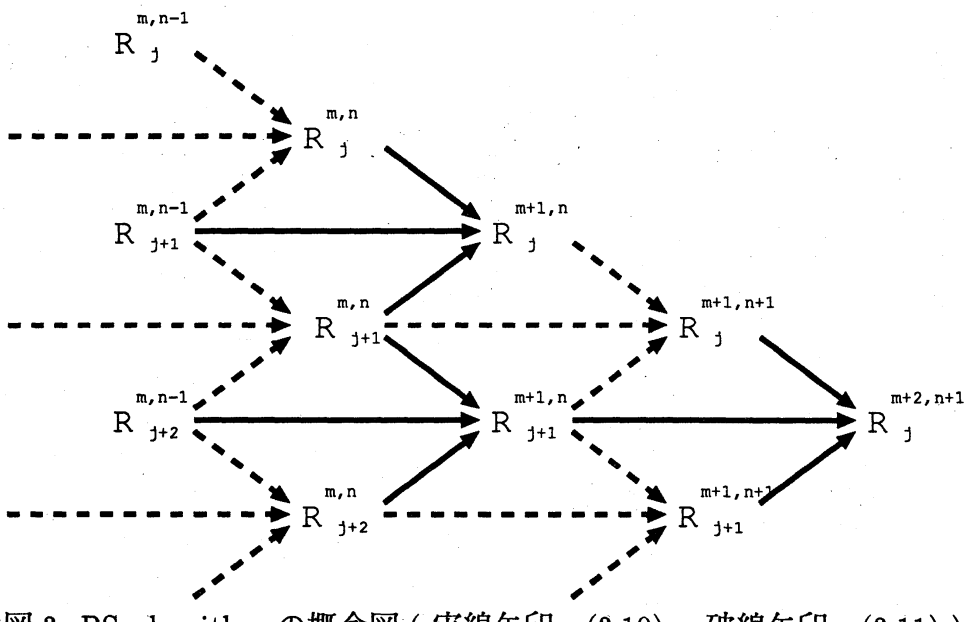

図 3: BS–algorithm の概念図 (実線矢印 : (3.10) 、破線矢印 : (3.11))

具体的には、(3.10) と (3.11) を交互に利用することで計算を進めるアルゴリズムになり、図 3 に示されるようになる。なお、最初に (3.10) と (3.11) のどちらを用いるかを先に決めておけばこのアルゴリズムは一つの漸化式で書き表され、単純化できる。

この段階で、x に具体的な値を代入することで補外が行なわれる。つまり、加速法になるのである。

さて、元の収束数列が $lim_{j\to\infty}x_j = 0$ で上の有理式で補間できると想定される場合は、$x \to 0$ として上のアルゴリズムに代入して$\frac{a_0}{b_0}$ を求めるものが加速法であり、**BS–加速法** と呼ばれるものである。

逆に元の収束数列が $lim_{j\to\infty}x_j = \infty$ で上の有理式で補間できると想定される場合 (ただし $m = n$ とする) は、$x \to \infty$ として上のアルゴリズムに代入して$\frac{a_m}{b_m}$ を求めるものが加速法であり、**ρ–algorithm** と呼ばれるものである。

4 補外式間に漸化式が存在する場合

補外式そのものに漸化式が存在し加速法が得られるという例として、指数補外および一般関数による補外によって加速法を得ている例を紹介する。

4.1 指数補外を用いる場合：Richardson 加速

補間・補外関数	指数関数 $y(x) = a_0 + a_1\lambda_1^x + a_2\lambda_2^x + \cdots$ (a_j:未知、λ_j:既知、$1 > \|\lambda_1\| > \|\lambda_2\|$)
補外式	a_0 への Richardson 補外 ($x \to \infty$) $R[x_1,\cdots,x_n] = \prod_{j=1}^{n-1}\left(\frac{1-\lambda_j^{x_{j+1}-x_j}s_-}{1-\lambda_j^{x_{j+1}-x_j}}\right)y_n$ ただし、$s_-y_j \equiv y_{j-1}$
漸化式 = 加速法	a_0 への **Richardson 加速** $R[x_1,\cdots,x_{n+1}] = \frac{R[x_2,\cdots,x_{n+1}] - \lambda_n^{x_{n+1}-x_n}R[x_1,\cdots,x_n]}{1-\lambda_n^{x_{n+1}-x_n}}$

表 3: 指数補外 (指数既知) に基づく加速法

表 3に指数補外 (指数既知) に基づく加速法の概要を表にしてある。

加速したい収束数列 $\{y_j\}_{j=1}^{\infty}$ が適当に標本点 $\{x_j\}_{j=1}^{\infty} : \lim_{j\to\infty} x_j = \infty$ をとると、十分おおきな j に対して標本点座標の指数関数式

$$y(x) = a_0 + a_1\lambda_1^x + a_2\lambda_2^x + \cdots \tag{4.1}$$

(ただし、a_j:未知、λ_j:既知、$1 > |\lambda_1| > |\lambda_2| > \cdots$とする)
で補外できると想定できる場合、n 項からなる上記の式が n 点の標本点 $\{x_j\}_{j=1}^{n}$ で $y(x_j) = y_j$ を満たすとしたとき、求められる a_0 を $R[x_1,\cdots,x_n]$ とすると、

$$R[x_1,\cdots,x_n] = \prod_{j=1}^{n-1}\left(\frac{1-\lambda_j^{x_{j+1}-x_j}s_-}{1-\lambda_j^{x_{j+1}-x_j}}\right)y_n \tag{4.2}$$

(ただし、$s_{-}y_j \equiv y_{j-1}$)
であり、この R の間には次の漸化式

$$R[x_1,\cdots,x_{n+1}] = \frac{R[x_2,\cdots,x_{n+1}] - \lambda_n^{x_{n+1}-x_n} R[x_1,\cdots,x_n]}{1-\lambda_n^{x_{n+1}-x_n}} \tag{4.3}$$

が成立する [7, pp.49–]。

この漸化式が **Richardson** 加速とよばれるものである。なお、実際には $x_j = j$ としたものを考えることが多い。

4.2 指数補外を用いる場合：ϵ–algorithm

補間・補外関数	指数関数 $y(x) = a_0 + a_1\lambda_1^x + a_2\lambda_2^x + \cdots$ (a_j, λ_j:未知、$1 > \lvert\lambda_1\rvert > \lvert\lambda_2\rvert > \cdots$)
補外式	a_0 への 補外 : Schmidt 変換 ($x \to \infty$)
漸化式 = 加速法	a_0 への ϵ–**algorithm**

表 4: 指数補外 (指数未知) に基づく加速法

表 4に指数補外 (指数未知) に基づく加速法の概要を表にしてある。

加速したい収束数列 $\{y_j\}_{j=1}^{\infty}$ が等間隔で並ぶ適当な標本点 $\{x_j\}_{j=1}^{\infty}$: $\lim_{j\to\infty} x_j = \infty$: $x_{j+1} - x_j =$Const. をとると、十分おおきな j に対して標本点座標の指数式

$$y(x) = a_0 + a_1\lambda_1^x + a_2\lambda_2^x + \cdots \tag{4.4}$$

(ただし、a_j, λ_j:未知、$1 > |\lambda_1| > |\lambda_2| > \cdots$とする)
で補外できると想定できる場合を考える。

$m+1$ 項からなる式 (自由度は $2m+1$)

$$S^m(x) \equiv a_0 + a_1\lambda_1^x + a_2\lambda_2^x + \cdots + a_m\lambda^m \tag{4.5}$$

が $2m+1$ 点の標本点 $\{x_j,\cdots,x_{j+2m}\}$ で $S^m(x_j) = y_j$ を満たすとしたとき、求められる a_0 を $R[y_j,\cdots,y_{j+2m}]$ とすると、これは

$$R[y_j,\cdots,y_{j+2m}] = \frac{H_{m+1}(y_{j+2m})}{H_m(\nabla^2 y_{j+2m})} \tag{4.6}$$

と表される。この表式を Schmidt 変換 [6] とよぶ。

ただし、H は Hankel 行列式で、

$$H_n(u_k) \equiv \begin{vmatrix} u_k & u_{k-1} & \cdots & u_{k-n+1} \\ u_{k-1} & u_{k-2} & & u_{k-n} \\ \vdots & & \ddots & \vdots \\ \vdots & & & \vdots \\ u_{k-n+1} & u_{k-n} & \cdots & u_{k-2n+2} \end{vmatrix} \tag{4.7}$$

と表される。
(なお、$\nabla u_k \equiv u_k - u_{k-1}$ である)

Schmidt 変換は、x_j が等間隔であるために $S^m(x_j) - a_0$ が m 階差分方程式の一般解の形をしていることを利用して導出される。具体的には、上記のことより適当な係数 C_k が存在して

$$\sum_{k=0}^{m} C_k\{S^m(x_{j-k}) - a_0\} = 0 \tag{4.8}$$

が成立するためこの式を $m+1$ 個連立させると、

$$\begin{pmatrix} S^m(x_{j+2m}) - a_0 & S^m(x_{j+2m-1}) - a_0 & \cdots & S^m(x_{j+m}) - a_0 \\ S^m(x_{j+2m-1}) - a_0 & S^m(x_{j+2m-2}) - a_0 & & S^m(x_{j+m-1}) - a_0 \\ \vdots & & \ddots & \vdots \\ S^m(x_{j+m}) - a_0 & S^m(x_{j+m-1}) - a_0 & \cdots & S^m(x_j) - a_0 \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_m \end{pmatrix} = \vec{0} \tag{4.9}$$

となる。ここで、

$$S^m(x_l) = y_l \qquad (l = j, j+1, \cdots, j+2m) \tag{4.10}$$

と近似することにより、上の式の中の $S^m(x_l)$ が y_l に、a_0 が $R[y_j, \cdots, y_{j+2m}]$ に置き換わって、

$$\begin{vmatrix} y_{j+2m} - R & y_{j+2m-1} - R & \cdots & y_{j+m} - R \\ y_{j+2m-1} - R & y_{j+2m-2} - R & & y_{j+m-1} - R \\ \vdots & & \ddots & \vdots \\ y_{j+m} - R & y_{j+m-1} - R & \cdots & y_j - R \end{vmatrix} = 0 \tag{4.11}$$

となり、これを変形することで Schmidt 変換が得られる。

一見しただけで分かるように、Schmidt 変換そのものによる $R[y_j, \cdots, y_{j+2m}]$ の計算量は非常に大きいものになる。

そこで考え出されたのが、次のϵ–**algorithm**[6] という加速法である。これは、初期値として $\varepsilon_j^{-1} = 0, \varepsilon_j^0 = y_j$ とすると、次の漸化式

$$\varepsilon_j^{n+1} = \varepsilon_{j-1}^{n-1} + \frac{1}{\varepsilon_j^n - \varepsilon_{j-1}^n} \tag{4.12}$$

に従えば、

$$\begin{aligned} \varepsilon_j^1 &= \frac{1}{\nabla y_j} && (4.13) \\ \varepsilon_j^{2k} &= \frac{H_{k+1}(y_j)}{H_k(\nabla^2 y_j)} = R[y_{j-2k}, \cdots, y_j] && (4.14) \\ \varepsilon_j^{2k+1} &= \frac{H_k(\nabla^3 y_j)}{H_{k+1}(\nabla y_j)} && (4.15) \end{aligned}$$

となるというアルゴリズムである。

このアルゴリズムの証明は Sylvester determinant identity を通して容易に示されるが、このアルゴリズムを導出できる原理は知られていない。

4.3 減衰関数補外を用いる場合 : E–algorithm

補間・補外関数	一般減衰関数 $y(x) = a_0 + a_1g_1(x) + a_2g_2(x) + \cdots$ (a_j:未知、$g_j(x)$:既知、$\lim_{x\to\infty} g_j(x) = 0$)
補外式	a_0 への 補外 : E- 変換 ($x \to \infty$)
漸化式 = 加速法	a_0 への **E–algorithm**

表 5: 減衰関数補外に基づく加速法

表 5に一般減衰関数補外に基づく加速法の概要を表にしてある。

加速したい収束数列 $\{y_j\}_{j=1}^{\infty}$ が適当な標本点 $\{x_j\}_{j=1}^{\infty} : \lim_{j\to\infty} x_j = \infty$ をとると、十分おおきな j に対して標本点座標の一般減衰関数による補間式

$$y(x) = a_0 + a_1g_1(x) + a_2g_2(x) + \cdots \tag{4.16}$$

(ただし、a_j:未知、$g_j(x)$:既知、$\lim_{x\to\infty} g_j(x) = 0$ とする)
で補外できると想定できる場合を考える。

$m+1$ 項からなる式(自由度は $m+1$) $S^m(x) \equiv a_0 + a_1g_1(x) + a_2g_2(x) + \cdots + a_mg_m(x)$ が $m+1$ 点の標本点 $\{x_j, \cdots, x_{j+m}\}$ で $S^m(x_j) = y_j$ を満たすとしたとき、求められる a_0 を $R[y_j, \cdots, y_{j+m}]$ とすると、これは

$$R[y_j, \cdots, y_{j+m}] = \frac{\begin{vmatrix} y_j & g_1(x_j) & \cdots & g_m(x_j) \\ y_{j+1} & g_1(x_{j+1}) & \cdots & g_m(x_{j+1}) \\ y_{j+2} & g_1(x_{j+2}) & \ddots & g_m(x_{j+2}) \\ \vdots & \vdots & & \vdots \\ y_{j+m} & g_1(x_{j+m}) & \cdots & g_m(x_{j+m}) \end{vmatrix}}{\begin{vmatrix} 1 & g_1(x_j) & \cdots & g_m(x_j) \\ 1 & g_1(x_{j+1}) & \cdots & g_m(x_{j+1}) \\ 1 & g_1(x_{j+2}) & \ddots & g_m(x_{j+2}) \\ \vdots & \vdots & & \vdots \\ 1 & g_1(x_{j+m}) & \cdots & g_m(x_{j+m}) \end{vmatrix}} \tag{4.17}$$

と表される。この表式を E–変換 [1] とよぶ。

E–変換は、補外近似により

$$R[y_j, \cdots, y_{j+m}] + a_1g_1(x_k) + \cdots a_mg_m(x_k) = y_k \quad (k = j, j+1, \cdots, j+m) \tag{4.18}$$

が成り立つとすることで、

$$\begin{pmatrix} R-y_j & g_1(x_j) & \cdots & g_m(x_j) \\ R-y_{j+1} & g_1(x_{j+1}) & \cdots & g_m(x_{j+1}) \\ R-y_{j+2} & g_1(x_{j+2}) & \ddots & g_m(x_{j+2}) \\ \vdots & \vdots & & \vdots \\ R-y_{j+m} & g_1(x_{j+m}) & \cdots & g_m(x_{j+m}) \end{pmatrix} \begin{pmatrix} 1 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} = \vec{0} \tag{4.19}$$

、すなわち

$$\begin{vmatrix} R-y_j & g_1(x_j) & \cdots & g_m(x_j) \\ R-y_{j+1} & g_1(x_{j+1}) & \cdots & g_m(x_{j+1}) \\ R-y_{j+2} & g_1(x_{j+2}) & \ddots & g_m(x_{j+2}) \\ \vdots & \vdots & & \vdots \\ R-y_{j+m} & g_1(x_{j+m}) & \cdots & g_m(x_{j+m}) \end{vmatrix} = 0 \tag{4.20}$$

が成立することから導かれる。

このE–変換も計算量が非常に大きいが、次のような漸化式が存在し、**E–加速法**として知られている。

これは初期値として $E_j^0 = y_j, g_{j,j}^0 = g_i(x_j)$ とすると次の連立漸化式

$$E_j^n = \frac{E_j^{n-1} g_{n,j+1}^{n-1} - E_{j+1}^{n-1} g_{n,j}^{n-1}}{g_{n,j+1}^{n-1} - g_{n,j}^{n-1}} \tag{4.21}$$

$$g_{i,j}^n = \frac{g_{i,j}^{n-1} g_{n,j+1}^{n-1} - g_{i,j+1}^{n-1} g_{n,j}^{n-1}}{g_{n,j+1}^{n-1} - g_{n,j}^{n-1}} \quad (i = n+1, n+2, \cdots) \tag{4.22}$$

に従えば、

$$E_j^n = R[x_j, x_{j+1}, \cdots, x_{j+n}] \tag{4.23}$$

であるというものである。

このアルゴリズムの証明は Sylvester determinant identity を用いての式変形の後、帰納法によって行なわれる。

5 最後に

加速法を補間・補外式の間に成り立つ漸化式を利用したものという視点から見た場合にどのように整理されるかを示した。

この視点から整理できる加速法はこれだけではなく、他にも Ford-Sidi alogrithm [3, 5] をはじめとして多くが存在するが、本文の目的は加速法の簡単な構造を分かりやすく示すことであるので最小限の例示にとどめた。

また、ϵ–アルゴリズムや E–アルゴリズムなどは正しいことは証明できるものの、その導出原理に相当するものが不明であり、構造が全体的に不透明である感が否めない。この点をクリアできれば加速法そのものがもっと分かりやすくなるであろう。

参考文献

[1] Claude BREZINSKI. A general extrapolation algorithm. *Numer. Math.*, Vol.35, pp.175–187, 1980.

[2] Claude BREZINSKI. Convergence acceleration methods: The past decade. *J. Comput. and Appl. Math.*, Vol.12 & 13, pp.19–36, 1985.

[3] William F. Ford and Avram Sidi. An algorithm for a generalization of the richardson extrapolation process. *SIAM J. Numer. Anal.*, Vol.24, No.5, pp.1212–1232, Oct. 1987.

[4] 長田 直樹 (NaokiOsada). 収束の加速法. RIMS 講究録, Vol.880, pp.28–43, Jul. 1994.

[5] Avram Sidi. On a generalization of the richardson extrapolation process. *Numer. Math.*, Vol.57, pp.365–377, 1990.

[6] Jet Wimp. *Sequence Transformations and Their Applications*, Vol. 154 of *MATHEMATICS IN SCIENCE AND ENGINEERING*. ACADEMIC PRESS, New York, 1981.

[7] 森正武, 室田一雄, 杉原正顯. 数値計算の基礎, 方法 1, 第 2 巻. 岩波書店, 〒101-02 東京都千代田区一ツ橋 2-5-5, May 1993.